```
MMM          MMM  TTTTTTTTTTTTTTT  HHH          HHH  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
MMM          MMM  TTTTTTTTTTTTTTT  HHH          HHH  RRRRRRRRRRRR    TTTTTTTTTTTTTTT  LLL
MMM          MMM  TTTTTTTTTTTTTTT  HHH          HHH  RRRRRRRRRRRR    TTTTTTTTTTTTTTT  LLL
MMMMMM    MMMMMM        TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMMMMM    MMMMMM        TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMMMMM    MMMMMM        TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM  MMM  MMM           TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM  MMM  MMM           TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM  MMM  MMM           TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  RRRRRRRRRRR         TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  RRRRRRRRRRRR        TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  RRRRRRRRRRRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR   RRR           TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR   RRR           TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR   RRR           TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR      RRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR      RRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR      RRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
```

```
UU      UU  VV      VV  XX      XX  PPPPPPPP      OOOOOO    WW      WW    CCCCCCCC          JJ
UU      UU  VV      VV  XX      XX  PPPPPPPP    OOOOOOOO    WW      WW   CCCCCCCCCC         JJ
UU      UU  VV      VV  XX      XX  PP      PP  OO      OO  WW      WW  CC                  JJ
UU      UU  VV      VV  XX      XX  PP      PP  OO      OO  WW      WW  CC                  JJ
UU      UU  VV      VV    XX  XX    PP      PP  OO      OO  WW      WW  CC                  JJ
UU      UU  VV      VV    XX  XX    PP      PP  OO      OO  WW      WW  CC                  JJ
UU      UU  VV      VV      XX      PPPPPPPP    OO      OO  WW      WW  CC                  JJ
UU      UU  VV      VV      XX      PPPPPPPP    OO      OO  WW      WW  CC                  JJ
UU      UU  VV      VV    XX  XX    PP          OO      OO  WW  WW  WW  CC          JJ      JJ
UU      UU  VV      VV    XX  XX    PP          OO      OO  WW  WW  WW  CC          JJ      JJ
UU      UU    VV  VV     XX      XX  PP          OO      OO  WWWW  WWWW  CC          JJ      JJ      ....
UU      UU    VV  VV     XX      XX  PP          OO      OO  WWWW  WWWW  CC          JJ      JJ      ....
UUUUUUUUUU      VV       XX      XX  PP            OOOOOO    WW      WW   CCCCCCCC    JJJJJJ        ....
UUUUUUUUUU      VV       XX      XX  PP            OOOOOO    WW      WW   CCCCCCCC    JJJJJJ        ....
```

```
LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

(2) 47 HISTORY ; Detailed Current Edit History
(3) 65 DECLARATIONS
(4) 101 OTS$POWCJ - COMPLEX*8 ** INTEGER*4

```
0000     1          .TITLE  UVX$POWCJ - COMPLEX ** INTEGER*4 power routine
0000     2          .IDENT  /1-006/          ; File UVXPOWCJ.MAR  Edit: SBL1006
0000     3  ;*****************************************************************************
0000     4  ;*                                                                           *
0000     5  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     6  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     7  ;*  ALL RIGHTS RESERVED.                                                     *
0000     8  ;*                                                                           *
0000     9  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    10  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    11  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    12  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    13  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    14  ;*  TRANSFERRED.                                                             *
0000    15  ;*                                                                           *
0000    16  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    17  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    18  ;*  CORPORATION.                                                             *
0000    19  ;*                                                                           *
0000    20  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    21  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    22  ;*                                                                           *
0000    23  ;*                                                                           *
0000    24  ;*****************************************************************************
0000    25  ;
0000    26  ;
0000    27  ; FACILITY: Language support library - user callable
0000    28  ;++
0000    29  ; ABSTRACT:
0000    30  ;
0000    31  ;       COMPLEX base to INTEGER*4 power.
0000    32  ;       Floating overflow can occur.
0000    33  ;       Undefined exponentiation can occur if
0000    34  ;       base = (0.,0.) and exp <=0
0000    35  ;
0000    36  ;--
0000    37  ;
0000    38  ; VERSION: 0
0000    39  ;
0000    40  ; HISTORY:
0000    41  ; AUTHOR:
0000    42  ;       Jonathan M. Taylor, 29-jun-77: Version 0
0000    43  ;
0000    44  ; Edit history for version 0
0000    45  ;
```

```
                                                J 14
0000     47          .SBTTL  HISTORY         ; Detailed Current Edit History
0000     48
0000     49
0000     50 ; Edit History for Version 0 of OTS$POWCJ
0000     51 ; 0-2 change MTH$$ERROR to MTH$$SIGNAL JMT 5-OCT-77
0000     52 ; 0-6   - Change FOR$FLAG_JACKET to MTH$FLAG_JACKET.  TNH 17-July-78
0000     53 ; 0-7   - Fix bug giving divide fault, or wrong results for
0000     54 ;         some negative powers. Also clean up comments.
0000     55 ;         SPR 20364  SBL 27-Oct-78
0000     56 ; 1-001 - Change version number to 1 and MTH__UNDEXP
0000     57 ;         to MTH$K_UNDEXP.  JBS 07-DEC-78
0000     58 ; 1-002 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000     59 ; 1-003 - Declare externals.  SBL 17-May-1979
0000     60 ; 1-004 - Use general mode addressing.  SBL 30-Nov-1981
0000     61 ; 1-005 - Removed all uses of D_floating instructions for Microvax.
0000     62 ;         JCW 23-FEB-83
0000     63 ; 1-006 - Correct name to OTS$POWCJ.  SBL 31-May-1983
```

```
                     0000     65          .SBTTL  DECLARATIONS
                     0000     66
                     0000     67 ;
                     0000     68 ; INCLUDE FILES:
                     0000     69 ;
                     0000     70
                     0000     71 ; EXTERNAL SYMBOLS:
                     0000     72 ;
                     0000     73
                     0000     74          .DSABL  GBL
                     0000     75          .EXTRN  MTH$$SIGNAL             ; Math error routine
                     0000     76          .EXTRN  OTS$DIVC                ; COMPLEX division routine
                     0000     77          .EXTRN  MTH$K_UNDEXP
                     0000     78
                     0000     79 ;
                     0000     80 ; MACROS:
                     0000     81 ;
                     0000     82
                     0000     83 ;
                     0000     84 ; EQUATED SYMBOLS:
                     0000     85 ;
                00000004     86          base = 4                        ; base input formal - by-value
                0000000C     87          exp = 12                        ; exponent input formal - by-value
                     0000     88
                     0000     89 ;
                     0000     90 ; OWN STORAGE:
                     0000     91 ;
                     0000     92
                     0000     93 ;
                     0000     94 ; PSECT DECLARATIONS:
                     0000     95 ;
                     0000     96
                00000000     97          .PSECT  _OTS$CODE PIC,SHR,LONG,EXE,NOWRT
                     0000     98                                          ; program section for OTS$ code
                     0000     99
```

```
                        .SBTTL  OTS$POWCJ - COMPLEX*8 ** INTEGER*4
                        ;**-
                        ;++
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       COMPLEX result = COMPLEX base ** INTEGER*4 exponent
                        ;       The COMPLEX result is given by:
                        ;
                        ;       base            exponent        result
                        ;
                        ;       any               >0            PRODUCT (base * 2**i) where
                        ;                                       i is each non-zero bit in
                        ;                                       exponent.
                        ;
                        ;       (0., 0.)          <=0           Undefined exponentiation.
                        ;
                        ;       not (0., 0.)       <0           PRODUCT (base * 2**i) where
                        ;                                       i is each non-zero bit in
                        ;                                       !exponent!.
                        ;
                        ;       not (0., 0.)       =0           (1.0, 0.0)
                        ;
                        ;                                                               ;**-
                        ;       Floating overflow can occur.
                        ;       Undefined exponentiation occurs if base is 0 and
                        ;       exponent is 0 or negative.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       Power.wfc.v = OTS$POWCJ (base.rfc.v,  exponent.rl.v)
                        ;
                        ; INPUT PARAMETERS:
                        ;       The base input parameter is standard FORTRAN COMPLEX.
                        ;       The exponent input parameter is a signed longword integer.
                        ;       Both input parameters are CALL BY VALUE.
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUT PARAMETERS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; FUNCTION VALUE:
                        ;
                        ;       COMPLEX base ** INTEGER*4 exponent
                        ;
                        ; SIDE EFFECTS:
                        ;
                        ;       SIGNALs SS$_ARITH with floating overflow hardware code if
                        ;       floating overflow.
                        ;       SIGNALs MTH$_UNDEXP (82 = ' UNDEFINED EXPONENTATION') if
                        ;       base is 0 and exponent is 0 or negative.
                        ;
                        ;--
```

```
                                007C  0000   159          .ENTRY  OTS$POWCJ, ^M<R2,R3,R4,R5,R6>
                                      0002   160                                               ; disable integer overflow
                          52    04 AC   7D   0002   161          MOVQ    base(AP), R2             ; R2/R3 gets COMPLEX base
                          54    0C AC   D0   0006   162          MOVL    exp(AP), R4              ; R4 = longword exponent
                                   03   18   000A   163          BGEQ    1$                       ; R4 = ! exponent !
                             54    54   CE   000C   164          MNEGL   R4, R4
                          0C 54    00   E5   000F   165 1$:      BBCC    #0, R4, EVEN             ; branch if even and clear low bit
                          50       52   7D   0013   166          MOVQ    R2, R0                   ; R0/R1 = initial result
                    54    54    FF 8F   9C   0016   167          ROTL    #-1, R4, R4              ; R4 = unsigned_exponent / 2
                                   5B   13   001B   168          BEQL    DONE                     ; done if exponent was 1
                                   2A   11   001D   169          BRB     SQUAR1                   ; else use rest of exponent
                                        001F   170
                                        001F   171 EVEN:
                                        001F   172
                                        001F   173 ;
                                        001F   174 ;        MOVD    #1, R0                   ; R0/R1 = initial result
                                        001F   175 ; now becomes
                                        001F   176 ;
                             51    00   D0   001F   177          MOVL    #0, R1                   ; Imaginary part of initial result is 0
                             50    08   50   0022   178          MOVF    #1, R0                   ; Real part of initial result is 1
                    54    54    FF 8F   9C   0025   179          ROTL    #-1, R4, R4              ; R4 = unsigned_exponent / 2
                                   1D   12   002A   180          BNEQ    SQUAR1                   ; branch if exponent not 0
                                   52   53   002C   181          TSTF    R2                       ; exponent was 0, text RP(base)
                                   48   12   002E   182          BNEQ    DONE                     ; done if non-0, answer is 1.0
                                   53   53   0030   183          TSTF    R3                       ; IP(base) better not be zero
                                   44   12   0032   184          BNEQ    DONE                     ; it isn't return 1.0
                                        0034   185
                                        0034   186 UNDEFINED:
                          50    01    0F   79   0034   187          ASHQ    #15, #1, R0              ; return R0 = reserved operand
                             7E    00'8F   9A   0038   188          MOVZBL  #MTH$K_UNDEXP, -(SP)     ; FORTRAN error number
                    00000000'GF    01   FB   003C   189          CALLS   #1, G^MTH$$SIGNAL        ; convert to 32-bit condition code
                                        0043   190                                               ; and SIGNAL MTH$_UNDEXP
                                   04   0043   191          RET
                                        0044   192
                                        0044   193 SQUAR:
                    54    54    FF 8F   78   0044   194          ASHL    #-1, R4, R4              ; R4 = |reduced exponent| / 2
                                        0049   195 ;
                                        0049   196 ; R2/R3 = square current base
                                        0049   197 ;
                                        0049   198 SQUAR1:
                          55    53    52   45   0049   199          MULF3   R2, R3, R5               ; R5 = tmp = RP(base)*IP(base)
                                52    52   44   004D   200          MULF    R2, R2                   ; R2 = RP(base)**2
                                53    53   44   0050   201          MULF    R3, R3                   ; R3 = IP(base)**2
                                52    53   42   0053   202          SUBF    R3, R2                   ; R2 = RP(base)**2 - IP(base)**2
                          53    55    55   41   0056   203          ADDF3   R5, R5, R3               ; R3 = 2*RP(base)*IP(base)
                                E7 54    E9   005A   204          BLBC    R4, SQUAR                ; branch if next exponent bit is 0
                                        005D   205 ;
                                        005D   206 ; R0/R1 = partial result * current power of base
                                        005D   207 ;
                          55    53    50   45   005D   208          MULF3   R0, R3, R5               ; R5 = tmp = RP(part) * IP(base)
                                50    52   44   0061   209          MULF    R2, R0                   ; R0 = RP(part) * RP(base)
                          56    53    51   45   0064   210          MULF3   R1, R3, R6               ; R6 = tmp = IP(part) * IP(base)
                                50    56   42   0068   211          SUBF    R6, R0                   ; R0=RP(part)*RP(base)-IP(part)*IP(base)
                                51    52   44   006B   212          MULF    R2, R1                   ; R1 = IP(part) * RP(base)
                                51    55   40   006E   213          ADDF    R5, R1                   ; R1=IP(part)*RP(base)+RP(part)*IP(base)
                    54    54    FF 8F   78   0071   214          ASHL    #-1, R4, R4              ; R4 = |reduced exponent| / 2
                                   D1   12   0076   215          BNEQ    SQUAR1                   ; loop if more exponent bits left
```

```
                                                N 14
                     0078   216 DONE:
                OC AC   D5   0078   217          TSTL    exp(AP)                 ; test exponent sign
                   18   18   007B   218          BGEQ    POWCJ                   ; done if positive
                   50   53   007D   219          TSTF    R0                      ; test RP(result)
                   04   12   007F   220          BNEQ    RECIP                   ; if non-0, OK to take reciprocal
                   51   53   0081   221          TSTF    R1                      ; RP(result) was 0, test IP(result)     ;**-
                   AF   13   0083   222          BEQL    UNDEFINED               ; undefined (0.0+0.0i) ** -n
                        0085   223 RECIP:
                7E 50   7D   0085   224          MOVQ    R0, -(SP)               ; second arg pair is divisor
                        0088   225 ;
                        0088   226 ;        MOVD    S^#1, -(SP)             ; push (1.,0.) on stack
                        0088   227 ; becomes
                        0088   228
                7E 00   D0   0088   229          MOVL    #0, -(SP)               ; push imaginary part on stack
                7E 08   50   008B   230          MOVF    S^#1, -(SP)             ; stack has (1.0 , 0.0)
00000000'GF     04   FB   008E   231          CALLS   #4, G^OTS$DIVC          ; R0/R1 = reciprocal
                        0095   232 POWCJ:                                                                          ;**-
                   04   0095   233          RET                             ; result in R0/R1
                        0096   234
                        0096   235          .END
```

```
BASE            = 00000004
DONE              00000078 R      01
EVEN              0000001F R      01
EXP             = 0000000C
MTH$$SIGNAL       ********   X    00
MTH$K_UNDEXP      ********   X    00
OTS$DIVC          ********   X    00
OTS$POWCJ         00000000 RG     01
POWCJ             00000095 R      01
RECIP             00000085 R      01
SQUAR             00000044 R      01
SQUAR1            00000049 R      01
UNDEFINED         00000034 R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
_OTS$CODE	00000096 (150.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.09	00:00:00.53
Command processing	130	00:00:00.48	00:00:03.15
Pass 1	71	00:00:00.60	00:00:02.94
Symbol table sort	0	00:00:00.00	00:00:00.03
Pass 2	56	00:00:00.46	00:00:01.64
Symbol table output	1	00:00:00.02	00:00:00.05
Psect synopsis output	2	00:00:00.02	00:00:00.13
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	296	00:00:01.69	00:00:08.57

The working set limit was 900 pages.
3207 bytes (7 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 13 non-local and 1 local symbols.
235 source lines were read in Pass 1, producing 11 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

```
MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:UVXPOWCJ/OBJ=OBJ$:UVXPOWCJ MSRC$:UVXPOWCJ/UPDATE=(ENH$:UVXPOWCJ)
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

OTSPOWHH
LIS

OTSPOWII
LIS

OTSPOWRJ
LIS

UUXPOWGG
LIS

OTSPOWGLU
LIS

OTSPOWHLU
LIS

OTSPOWHJ
LIS

UUXPOWCJ
LIS

UUXPOWRR
LIS

OTSPOWLUL
LIS

OTSPOWRR
LIS

OTSPOWJJ
LIS

UUXEXP
LIS

UUXGSINCO
LIS

OTSPOWRLU
LIS